Open-Source Software Practice

Lab 07. Testing and Publishing

Instructor: Jaemin Jo (조재민, jmjo@skku.edu)
Interactive Data Computing Lab (*IDCLab*),
College of Computing and Informatics,
Sungkyunkwan University

Goals

1. Implement the “stat” command
2. Unit testing
3. Automate the tests via GitHub Actions
4. Publish the stat package to NPM

Let’s Practice – 1 (*stat*)

1. Create an empty Git repository
2. EditorConfig, README.md, .gitignore, …
3. `npm init`
4. Write the code in the next slides

main.js

```javascript
const lib = require("./lib");

if (process.argv.length <= 3) {
    console.log("Insufficient parameter!");
    process.exit(1);
}

let command = process.argv[2];

let numbers = process.argv
    .slice(3, process.argv.length)
    .map((n) => parseFloat(n));

if (numbers.some((n) => isNaN(n))) {
    console.log("Some arguments are not numbers!");
    process.exit(1);
}

let result;
switch (command) {
    case "sum":
        result = lib.sum(numbers);
        break;
    case "avg":
        result = lib.avg(numbers);
        break;
    case "max":
        result = lib.max(numbers);
        break;
    default:
        console.log("Wrong command!");
        process.exit(1);
}

console.log(result);
```

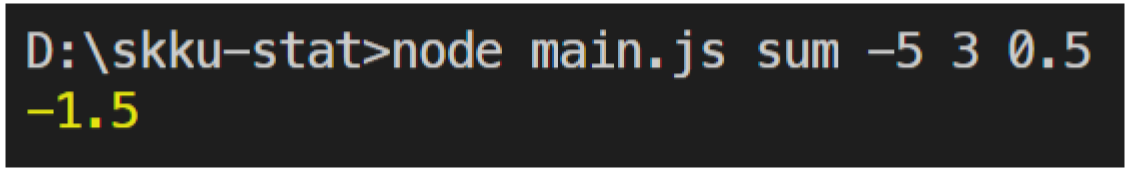
lib.js

```
function sum(numbers) {
  let s = 0;
  for (let i = 0; i < numbers.length; i++) s += numbers[i];
  return s;
}

function avg(numbers) {
  return sum(numbers) / numbers.length;
}

function max(numbers) {
  let m = numbers[0];
  for (let i = 1; i < numbers.length; i++) if (m < numbers[i]) m = numbers[i];
  return m;
}

exports.sum = sum;
exports.avg = avg;
exports.max = max;
```

(better) lib.js

```
function sum(numbers) {
  return numbers.reduce((prev, curr) => prev + curr, 0);
}

function avg(numbers) {
  return sum(numbers) / numbers.length;
}

function max(numbers) {
  return numbers.reduce((max, curr) => (max > curr ? max : curr), numbers[0]);
}

module.exports = {
  sum,
  avg,
  max,
};
```

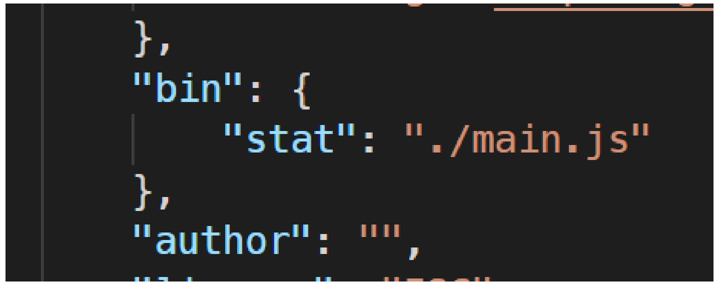
Let’s Practice – 1 (*stat*)



```
D:\skku-stat>node main.js sum -5 3 0.5
-1.5
```

Linking a Command

- Let’s run our script as a command.



```
},
"bin": {
    "stat": "./main.js"
},
"author": "",
```

1. Add *#!/usr/bin/env node* to the first line of *main.js*.
2. Link *main.js* to a command *stat* in *package.json*.
3. `npm link`
4. You should be able to run the “stat” command on a terminal.
5. `npm unlink`
6. `push everything`

Testing

- We manually tested the program by giving some small inputs.

- But what if the program gets bigger?
  - More components
  - Complex test scenarios
  - Heterogenous environments (e.g., Node.js versions)
  - Different developers

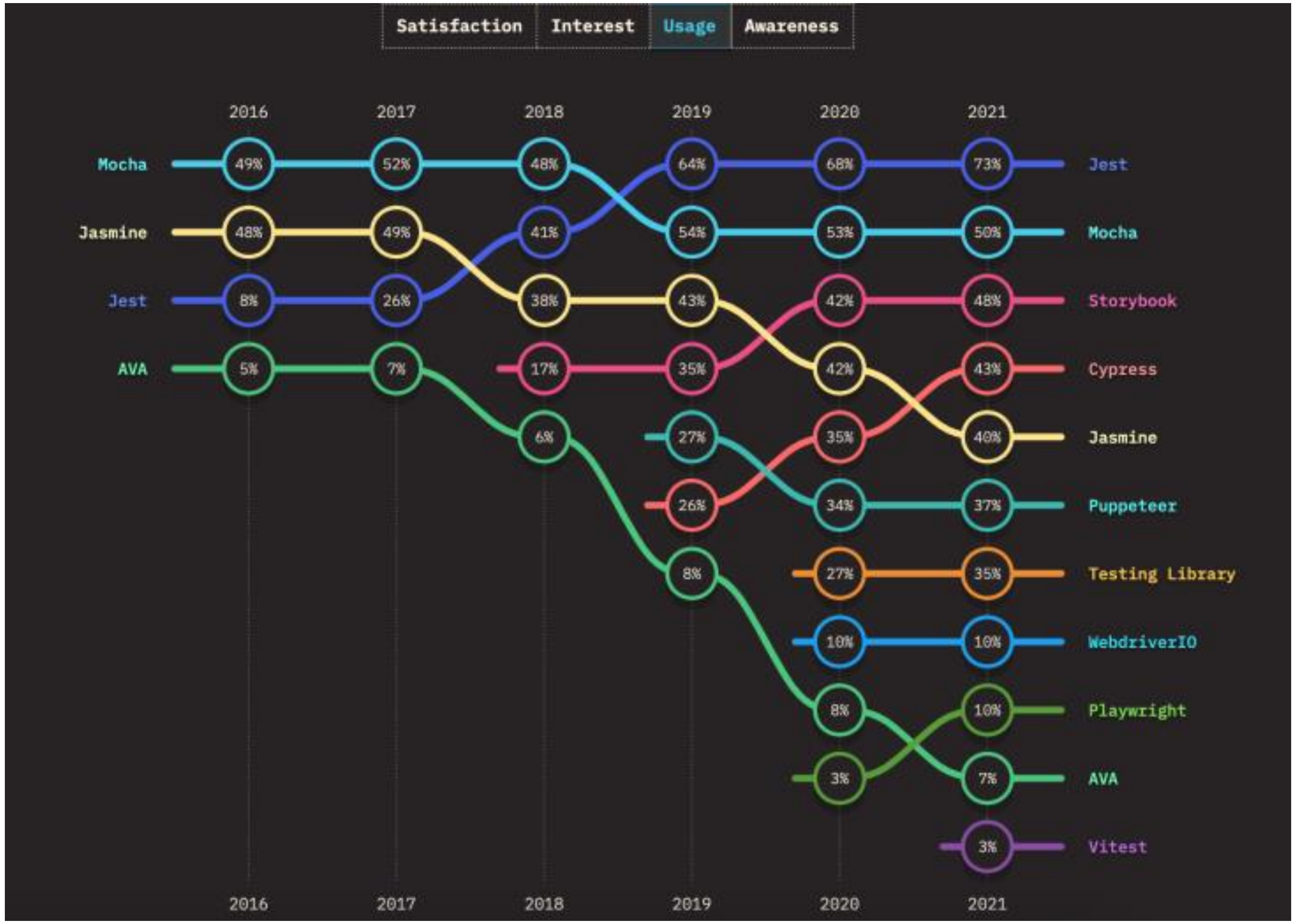
- We need to automate the test.

Unit Testing in Node

- There are a lot of frameworks that you can use to do unit testing in Node.
  - Jest, mocha, chai, jasmine, …
  - We will use Jest.
  - https://jestjs.io/

- `npm install jest --save-dev`
  - Note that we set the --save-dev flag for installation.
  - Jest is **not** required to run our program (ours does not have external dependencies!) but is needed to test our program as part of development.

Unit Testing in Node



--save-dev

	npm install <name>	npm install <name> --save-dev
Meaning	Required package	Only required for development
Where the dependency is listed	“dependencies” in *package.json*	“devDependencies” in *package.json*
Who concerns?	All users (end-user + developers)	Only developers want to extend your package

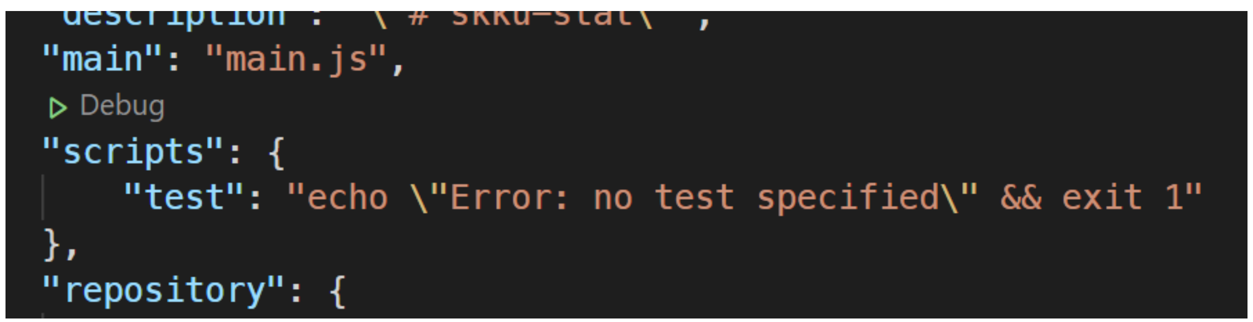
Let’s Practice – 2 (writing tests)

- Create *lib.test.js* and write the code on the right
  - Load the file that is tested (*lib.js*).
  - Run some functions with fixed input.
  - Check if the result is the same as the expected output.

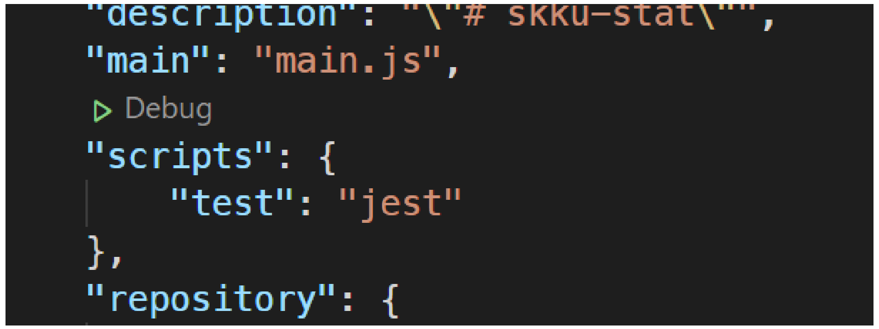
```
const { test, expect } = require("@jest/gl
obals");
const lib = require("./lib");

test("sum([1, 2]) should be 3", () => {
    expect(lib.sum([1, 2])).toBe(3);
});

test("avg([-5, 5]) should be 0", () => {
    expect(lib.avg([-5, 5])).toBe(0);
});

test("max([0, 3, 2]) should be 3", () => {
    expect(lib.max([0, 3, 2])).toBe(3);
});
```

Let’s Practice – 2 (writing tests)

- After you added a test, run it via `npm test`.
- But you will see an error since Jest is not linked to the command.
- Open *package.json* and set “test” under “scripts” to “jest”.



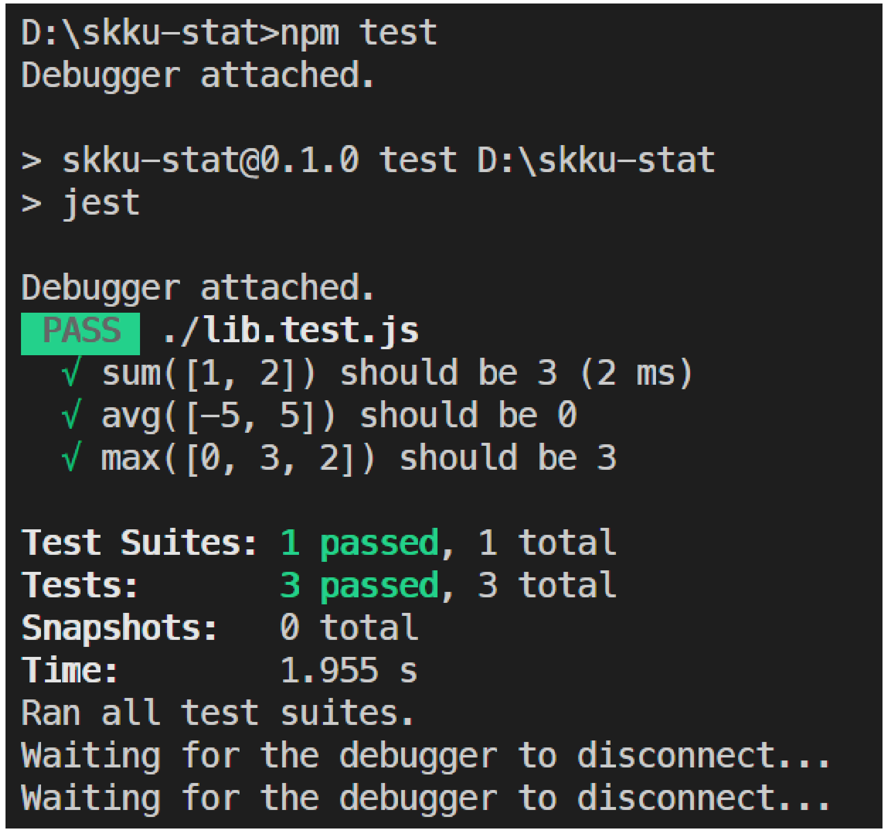
```
"description": "\"# skku-stat\"",
"main": "main.js",
▷ Debug
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
},
"repository": {
```



```
"description": "\"# skku-stat\"",
"main": "main.js",
▷ Debug
"scripts": {
    "test": "jest"
},
"repository": {
```

Let’s Practice – 2 (writing tests)

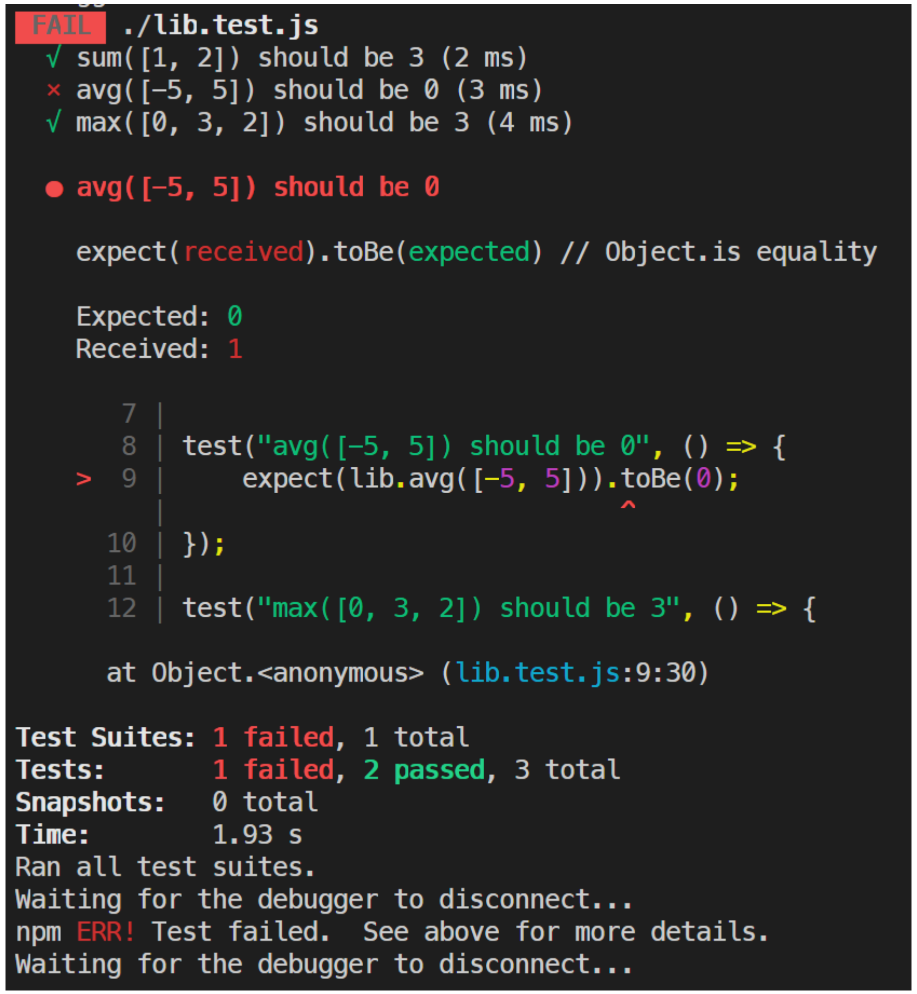
- `npm test`
- Congrats! You just finished your first unit testing.
- What happens if it fails?



```
D:\skku-stat>npm test
Debugger attached.

> skku-stat@0.1.0 test D:\skku-stat
> jest

Debugger attached.
 PASS  ./lib.test.js
  √ sum([1, 2]) should be 3 (2 ms)
  √ avg([-5, 5]) should be 0
  √ max([0, 3, 2]) should be 3

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        1.955 s
Ran all test suites.
Waiting for the debugger to disconnect...
Waiting for the debugger to disconnect...
```

Running Tests

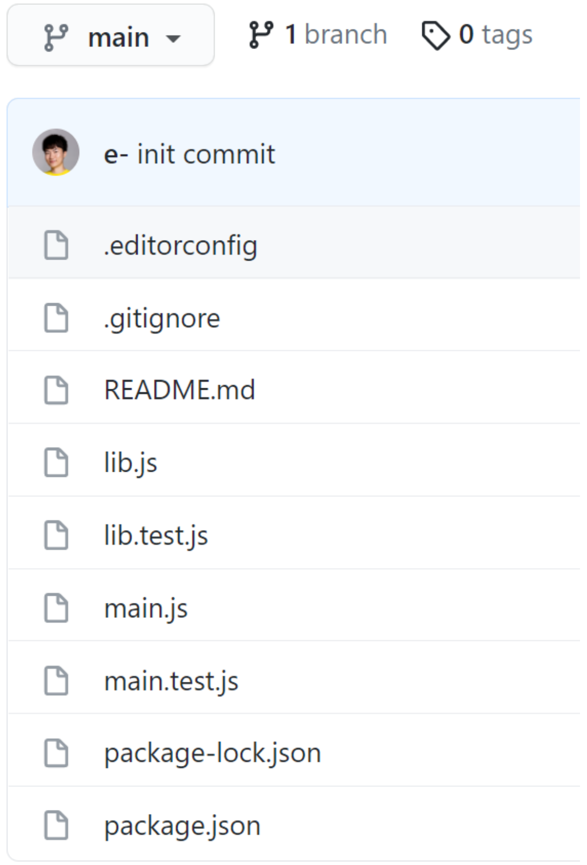
- If your test fails, you will see:



```
 FAIL  ./lib.test.js
  √ sum([1, 2]) should be 3 (2 ms)
  × avg([-5, 5]) should be 0 (3 ms)
  √ max([0, 3, 2]) should be 3 (4 ms)

  ● avg([-5, 5]) should be 0

    expect(received).toBe(expected) // Object.is equality

    Expected: 0
    Received: 1

       7 |
       8 | test("avg([-5, 5]) should be 0", () => {
    >  9 |     expect(lib.avg([-5, 5])).toBe(0);
         |                              ^
      10 | });
      11 |
      12 | test("max([0, 3, 2]) should be 3", () => {

      at Object.<anonymous> (lib.test.js:9:30)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 2 passed, 3 total
Snapshots:   0 total
Time:        1.93 s
Ran all test suites.
Waiting for the debugger to disconnect...
npm ERR! Test failed.  See above for more details.
Waiting for the debugger to disconnect...
```

Git Actions

- Let’s push the code to Github.
- It would be awesome if unit tests are automatically run every time we update the code.
- This can be automated by GitHub Actions.
- Go to the Action tab in your repo.

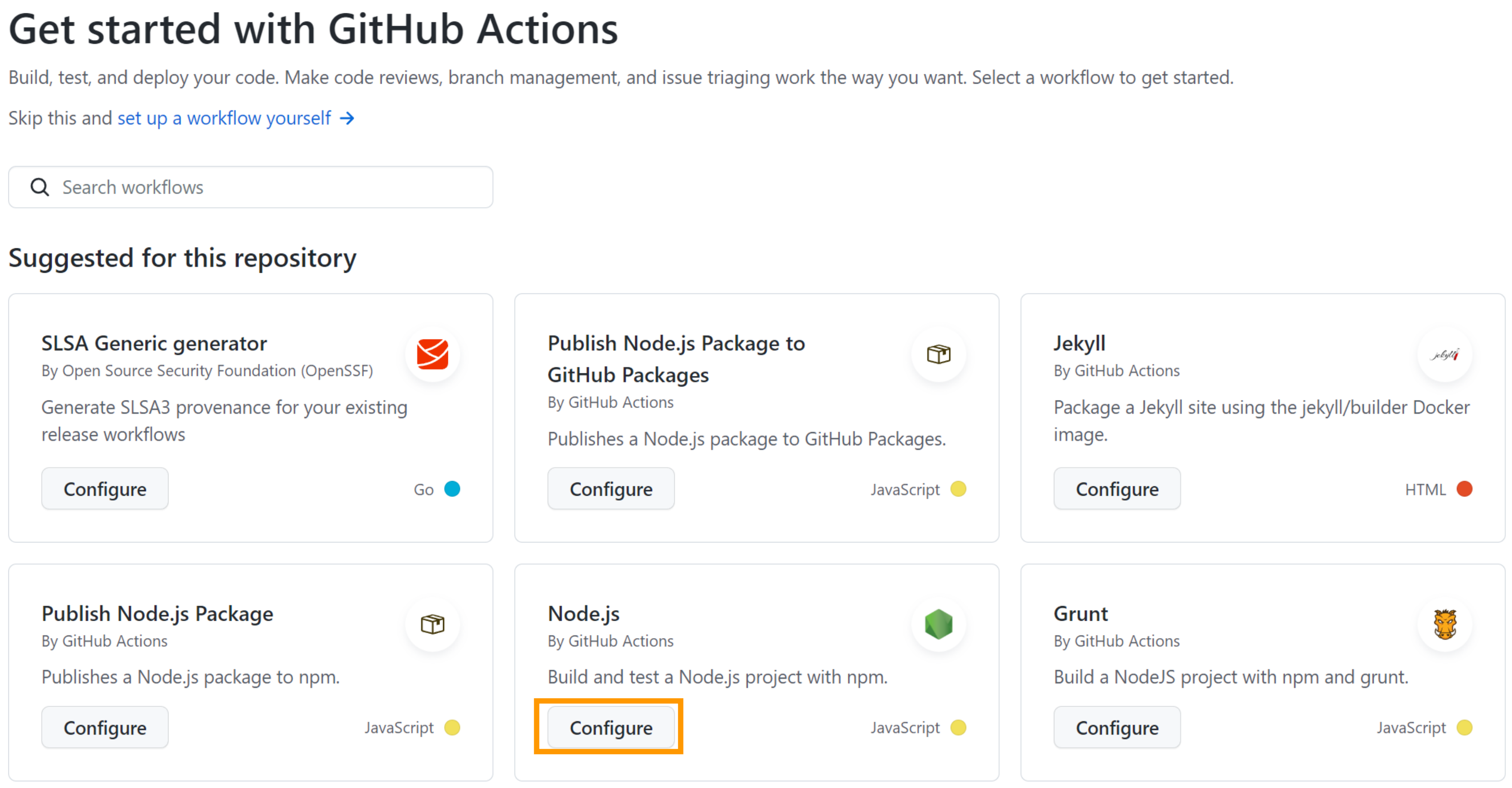


e- init commit

.editorconfig
.gitignore
README.md
lib.js
lib.test.js
main.js
main.test.js
package-lock.json
package.json

Let’s Practice – 3 (automation)

- Click on “Set up this workflow” under Node.js



Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and set up a workflow yourself →

Search workflows

Suggested for this repository

SLSA Generic generator
By Open Source Security Foundation (OpenSSF)
Generate SLSA3 provenance for your existing release workflows
Configure — Go

Publish Node.js Package to GitHub Packages
By GitHub Actions
Publishes a Node.js package to GitHub Packages.
Configure — JavaScript

Jekyll
By GitHub Actions
Package a Jekyll site using the jekyll/builder Docker image.
Configure — HTML

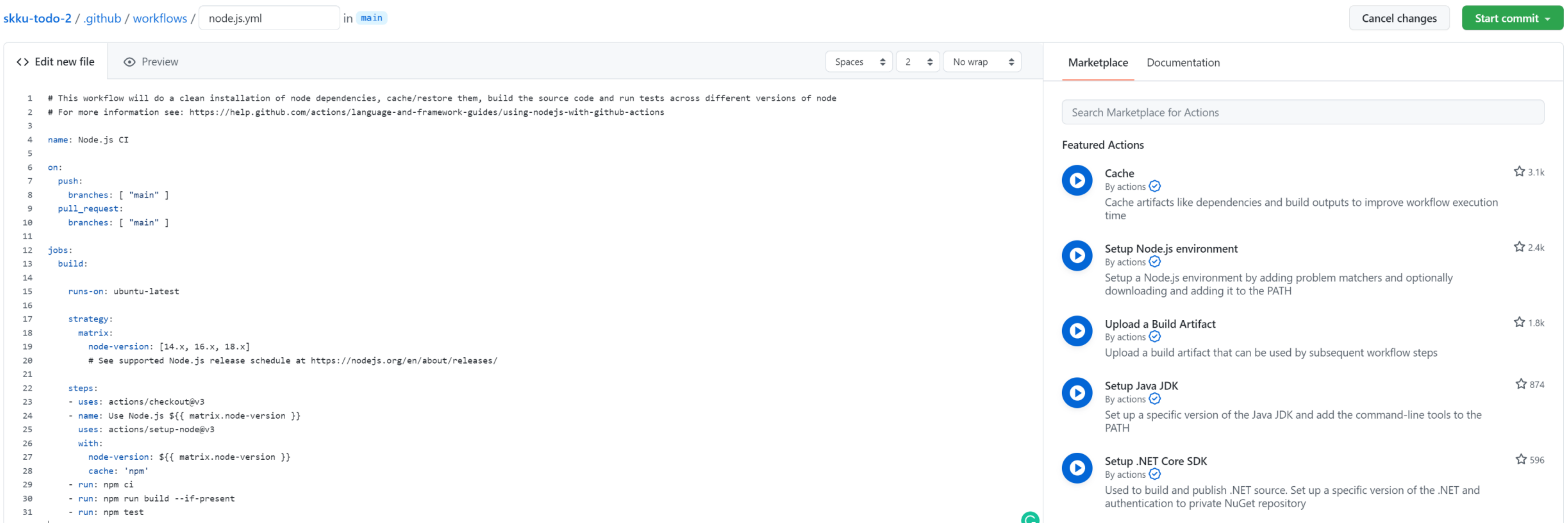
Publish Node.js Package
By GitHub Actions
Publishes a Node.js package to npm.
Configure — JavaScript

Node.js
By GitHub Actions
Build and test a Node.js project with npm.
Configure — JavaScript

Grunt
By GitHub Actions
Build a NodeJS project with npm and grunt.
Configure — JavaScript

Let’s Practice – 3 (automation)

- You can configure the workflow, but let’s use the default.
- Press “Start commit” and make a commit.


skku-todo-2 / .github / workflows / node.js.yml in main
Cancel changes
Start commit
Edit new file
Preview
Spaces
2
No wrap
Marketplace
Documentation
Search Marketplace for Actions

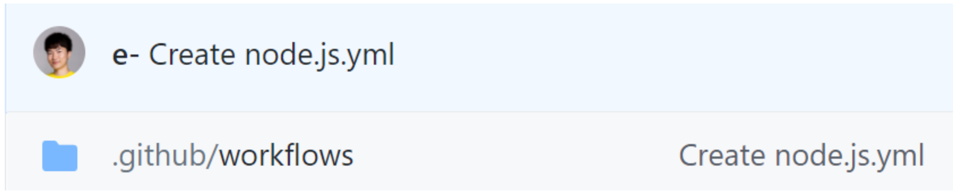
```yaml
# This workflow will do a clean installation of node dependencies, cache/restore them, build the source code and run tests across different versions of node
# For more information see: https://help.github.com/actions/language-and-framework-guides/using-nodejs-with-github-actions

name: Node.js CI

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:

    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [14.x, 16.x, 18.x]
        # See supported Node.js release schedule at https://nodejs.org/en/about/releases/

    steps:
    - uses: actions/checkout@v3
    - name: Use Node.js ${{ matrix.node-version }}
      uses: actions/setup-node@v3
      with:
        node-version: ${{ matrix.node-version }}
        cache: 'npm'
    - run: npm ci
    - run: npm run build --if-present
    - run: npm test
```

Featured Actions

Action	Publisher	Description	Stars
Cache	By actions	Cache artifacts like dependencies and build outputs to improve workflow execution time	3.1k
Setup Node.js environment	By actions	Setup a Node.js environment by adding problem matchers and optionally downloading and adding it to the PATH	2.4k
Upload a Build Artifact	By actions	Upload a build artifact that can be used by subsequent workflow steps	1.8k
Setup Java JDK	By actions	Set up a specific version of the Java JDK and add the command-line tools to the PATH	874
Setup .NET Core SDK	By actions	Used to build and publish .NET source. Set up a specific version of the .NET and authentication to private NuGet repository	596

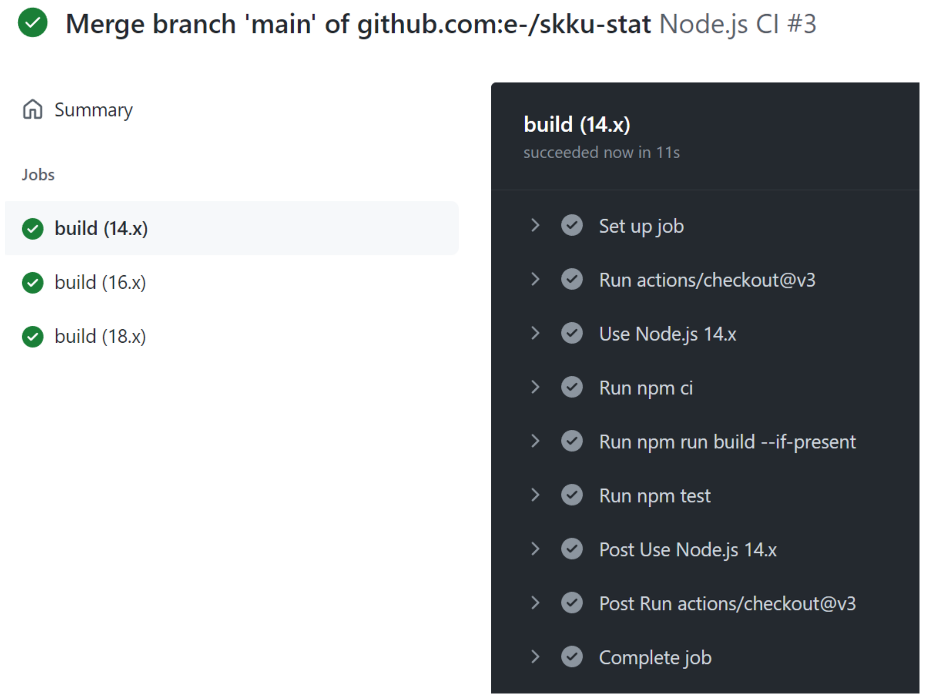
Let’s Practice – 3 (automation)

- A configuration file is added to your repo.


e- Create node.js.yml

.github/workflows Create node.js.yml

- Each time you make a new commit, Git Actions will run the tests on different versions of Node.js.
  - You can view the status in the Action tab.


Merge branch 'main' of github.com:e-/skku-stat Node.js CI #3

Summary

Jobs

build (14.x)
build (16.x)
build (18.x)

build (14.x)
succeeded now in 11s

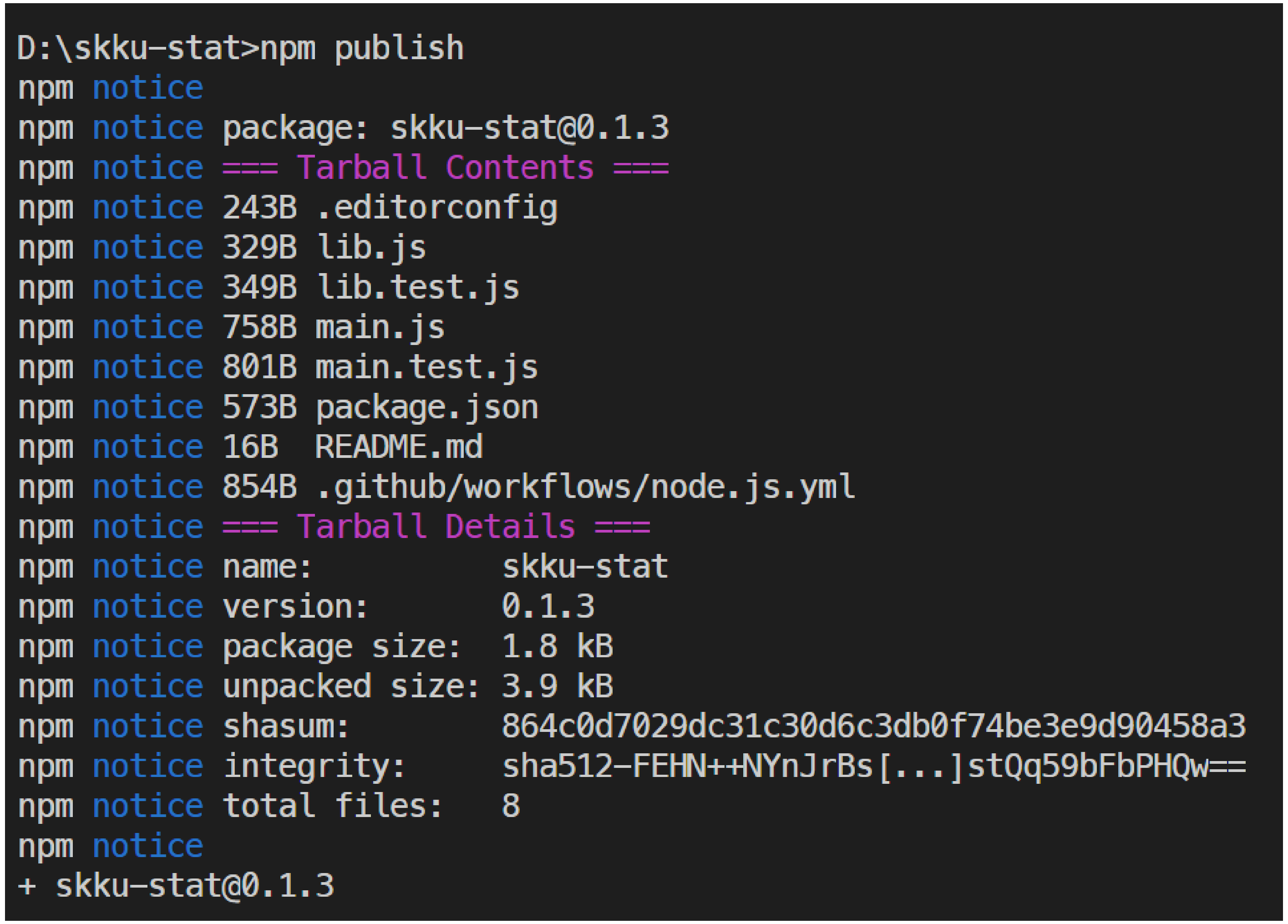
```
Set up job
Run actions/checkout@v3
Use Node.js 14.x
Run npm ci
Run npm run build --if-present
Run npm test
Post Use Node.js 14.x
Post Run actions/checkout@v3
Complete job
```

Let’s Practice – 4 (publish)

- Let’s publish our project to NPM: `npm publish`
- Change the project name since *“skku-stat”* is already taken.

- You will see errors since you are not logged in.
  - Sign up NPMJS (https://www.npmjs.com/).
  - Finish email verification. If not, your project will not be published.
  - `npm login` (password?)
  - `npm publish`
  - Don’t forget that you need to increment the version of your project (in *package.json*) at least by 0.0.1 each time you publish.
  - We call this “bump version”.

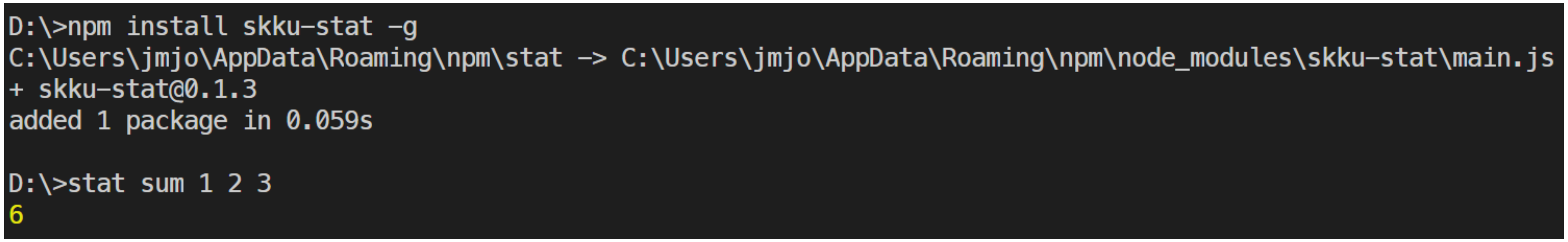
Let’s Practice – 4 (publish)



```
D:\skku-stat>npm publish
npm notice
npm notice package: skku-stat@0.1.3
npm notice === Tarball Contents ===
npm notice 243B .editorconfig
npm notice 329B lib.js
npm notice 349B lib.test.js
npm notice 758B main.js
npm notice 801B main.test.js
npm notice 573B package.json
npm notice 16B  README.md
npm notice 854B .github/workflows/node.js.yml
npm notice === Tarball Details ===
npm notice name:          skku-stat
npm notice version:       0.1.3
npm notice package size:  1.8 kB
npm notice unpacked size: 3.9 kB
npm notice shasum:        864c0d7029dc31c30d6c3db0f74be3e9d90458a3
npm notice integrity:     sha512-FEHN++NYnJrBs[...]stQq59bFbPHQw==
npm notice total files:   8
npm notice
+ skku-stat@0.1.3
```

Let's Practice – 4 (publish)

- Let's check out if it's been published well by installing it via NPM.
- `npm install <your_project_name> -g`
- `<your_project_name>` is the project name in *package.json*.

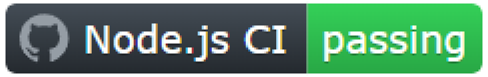


```
D:\>npm install skku-stat -g
C:\Users\jmjo\AppData\Roaming\npm\stat -> C:\Users\jmjo\AppData\Roaming\npm\node_modules\skku-stat\main.js
+ skku-stat@0.1.3
added 1 package in 0.059s

D:\>stat sum 1 2 3
6
```

Further Questions

- How can we add a “CI passing” badge to README.md file?
  - CI = continuous integration

- Can we automatically publish a package to NPM by using GitHub Actions?


Node.js CI
passing